Travis-CI - Autodeploy to k8s

# Script

#!/bin/bash -ev

# TravisCI Autodeploy to Kubernetes Script

# Ensure all required environment variables are present

if [ -z "$KUBERNETES\_CLUSTER\_CERTIFICATE" ] || \

[ -z "$KUBERNETES\_SERVER" ] || \

[ -z "$KUBERNETES\_SERVICE\_ACC\_TOKEN" ]; then

>&2 echo 'Required variable unset, docker build and deploy failed'

exit 1

fi

# Generate latest docker image name

container\_version=$(git rev-parse --short=8 HEAD)

echo : "

Travis-CI autodeploy to kubernetes script

Repo: $TRAVIS\_REPO\_SLUG

Image Version: $container\_version

"

# Update kube deployment files with new container version

find ./kube/ -type f | xargs sed -i "s/CONTAINER\_VERSION/$container\_version/g"

# Install kubernetes cli and add to path

curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

# Create cerifiticate file

echo "$KUBERNETES\_CLUSTER\_CERTIFICATE" | base64 --decode > cert.crt

# Apply updated kubernetes application config

kubectl --kubeconfig=/dev/null \

--certificate-authority=cert.crt \

--server=$KUBERNETES\_SERVER \

--token=$KUBERNETES\_SERVICE\_ACC\_TOKEN \

apply -f ./kube/

# Detail

#### Automatically apply updated Kubernetes config to control plane using Travis-CI

Add script to project in suitable scripts directory:

```

./scripts/travis-ci/travis-ci-autodeploy-kube.sh

```

Add the ```travis-ci-autodeploy-kube.sh script``` as a script to the travis-ci deploy hook in ```travis.yml```:

```yaml

deploy:

provider: script

script: bash ./scripts/travis-ci/travis-ci-autodeploy-kube.sh

on:

branch: master

```

In the Kubernetes deployment create and apply a ServiceAccount object and bind a role to it, allowing limited deploy access to the cluster. For example:

Account: ```cicd-service-account.yml```

```yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: cicd

namespace: default

```

Role: ```cicd-role.yml```

```yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: cicd

rules:

- apiGroups: ["", "apps", "batch", "extensions", "storage.k8s.io"]

resources: ["deployments", "services", "replicasets", "pods", "jobs", "cronjobs", "storageclasses", "persistentvolumeclaims"]

verbs: ["\*"]

```

Binding: ``` cicd-role-binding.yml

```yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: cicd

subjects:

- kind: ServiceAccount

name: cicd

namespace: default

roleRef:

kind: ClusterRole

name: cicd

apiGroup: rbac.authorization.k8s.io

```

Ensure the required environment variables are set in the build. For secret credentials, add to build directly on [travis-ci.com](https://travis-ci.com), or by using [Travis secure values](https://docs.travis-ci.com/user/encryption-keys/#usage) (see notes on max length of secure credentials).

- KUBERNETES\_CLUSTER\_CERTIFICATE - Can be found (base64 encoded) in the .kube/config file on the Kubernetes server.

- KUBERNETES\_SERVER - Can also be found in the ```.kube/config``` file on the Kubernetes server. Note the IP address may need substituting for your domain name.

- KUBERNETES\_SERVICE\_ACC\_TOKEN - After the above service account has been created, run ```kubectl get secret $(kubectl get secrets | grep cicd-token | awk '{print $1}') -o jsonpath='{.data.token}' | base64 --decode``` on kubernetes server to get the service account access token.

Before deploying, the script will also search and replace and references of 'CONTAINER\_VERSION' in config files contained in ```./kube/``` with the first characters of the git commit. This provides specific container deployment and identification, allowing for quick rollbacks if required.

##### Notes

- The auto-deploy script assumes you have tagged your build image name with the first 8 characters of the git commit hash it was built from.

- Max length of Travis-CI encrypted values seems to be around 128 bytes, therefore the KUBERNETES\_CLUSTER\_CERTIFICATE and KUBERNETES\_SERVICE\_ACC\_TOKEN will have to be added directly to Travis-CI using either their web app or ```travis set env```.

- When adding environment variables as Travis-CI secure values they will only be available in pull requests created by trusted users. Therefore, if an outside user makes a pull request to your repository, they will not have access to your access token and the deployment will fail.

- Depending on when your travis-ci account was made, secret may require encrypting with the ``` --com``` flag. See [this stackoverflow]( https://stackoverflow.com/questions/58152790/why-does-my-secure-travis-ci-environment-variable-not-work) for more details.